On the use of Java Server Side Technologies for the Design of Dynamically Redeployable MMOGs

Bruno Van Den Bossche, Bart De Vleeschauwer, Tom Verdickt,
Filip De Turck, Bart Dhoedt and Piet Demeester
Ghent University - IBBT - IMEC
Department of Information Technology
Gaston Crommenlaan 8 bus 201 9050 Gent, Belgium
Email: {Bruno.VanDenBossche|Bart.DeVleeschauwer|Tom.Verdickt}@intec.UGent.be

Abstract – *The number of players participating in Massively Multiplayer Online games is going beyond the millions, an efficient software architecture is needed to manage these huge digital worlds and cope with the dynamically changing loads at the server side. These virtual environments are more and more dynamic and sudden increases in player density in a part of the world have an impact on the load of the server responsible for that section of the virtual world.*
In this paper we discuss a novel way to support this kind of application by dividing the virtual world into smaller parts, called microcells. These microcells can be reassigned dynamically to the available servers to optimize the load distribution and thus improve the game performance. In order to implement this architecture we evaluated JAIN SLEE and J2EE as a middleware platform for developing an MMOG, focusing on the low latency behavior of the application server.

Keywords: Massively Multiplayer Online Games, Distributed Architecture, Performance Evaluation

1 Introduction

With the widespread availability of broadband Internet access, online entertainment is getting more popular by the day. Recent years have seen an enormous increase in the number of players engaging in Massively Multiplayer Online Games (MMOGs). These games offer the players a huge virtual world in which they can freely interact with tens of thousands of other players, build new identities and go on quests to improve their ranking. Examples of these games include World of Warcraft [1] and Second Life [2], the former currently has over 6,000,000 active users with peaks of over 500,000 players interacting at the same time in the digital world. With a new generation of game consoles [3]–[5] on the horizon, all boasting broadband capabilities, this type of game will no doubt enjoy an even greater popularity in the future.
To cope with the large worlds and huge numbers of interacting players, an efficient architecture is required. Due to the massive scale of these games a single server is not able to support the virtual world, home to thousands of players. As a result the MMOGs distribute the load among a grid or cluster of servers. In World of Warcraft the world is replicated in a number of "realms". These realms are separate entities and it is impossible for players to interact with players from other realms. Taking this approach, it is possible to distribute the load over a number of clusters which can even be geographically dispersed. An alternative approach is taken by Second Life, where the virtual world is divided into different cells and each cell is hosted on its own server. These cells communicate with their neighbors and the players populating them and manage all objects located within their boundaries. To prevent an overloaded system, it is necessary to either limit the number of players which can enter a realm, or each server should be provisioned so it can cope with any load a single cell can generate. While the first approach limits the freedom of the users, the second approach causes an inefficient use of the available resources.
In order to handle the dynamic characteristics of MMOGs more efficiently, we developed a new architecture to dynamically redistribute the load over a set of servers by dividing the virtual world into a large number of smaller parts, called microcells, and distributing these microcells among the available servers. A more detailed description of this concept is given in section 2 followed by an architectural overview of the Dynamic Microcell architecture in section 3. Section 4 list the requirements that need to be met for this type of architecture and section 5 presents a functional overview of both J2EE and JAIN SLEE. In sections 6,7 and 8 the use case used for benchmarking, the test setup and the obtained results are presented. Conclusions and future work are given in section 9.

2 Microcells

The concept of dividing a virtual world into smaller parts, called cells, is currently already in use and might be considered obvious. Each cell consists of a part of the world and is responsible for its own contents, the communication with neighboring cells and the players populating it. An example of such a distribution is shown in figure 1 where the world is divided in four cells, equal in size.
This is the concept used in Second Life where the world is divided in squares each assigned to its own server. An

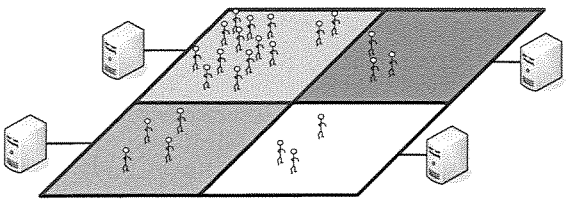


Fig. 1. Distribution of the virtual world into multiple cells in order to distribute the load.

important requirement of this architecture is that the cell-concept is completely transparent. Players should not need to know in which cell they currently reside or if they are crossing a border. This implies that players should be able to "see" across borders and even interact with other players in another cell (e.g. throw rocks at them). Depending on the landscape or by using (in game) binoculars it might even be possible to see across multiple cells.

Using multiple cells allows the load of the world to be distributed across multiple servers, but depending on the load caused by each cell, it is still possible to overload one server when too many players enter one single cell. As the freedom of players is less and less restricted in each generation of MMOGs this is not an unlikely event. For example when two players were to be wedded and throw a huge party where everyone is invited, this could result into one severely overloaded server and three servers idling.

To cope with these highly dynamic virtual worlds it should be possible to dynamically reassign the load to the available servers. To make this possible, extra requirements need to be met. The cells have to be dynamic, such that is possible to redeploy any given cell onto another server while the application is running and this completely transparent to the players. As we would like to reassign the load by dynamically redeploying the cells, the cells themselves need to be smaller as well. Therefore we propose to divide the world into smaller "microcells" which can be more easily reassigned to other servers. An overview of this concept is shown in figure 2.

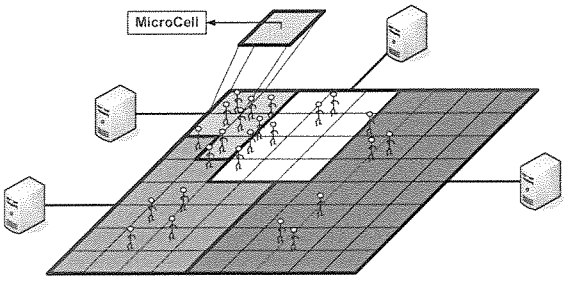


Fig. 2. Distribution of the virtual world into a large amount of smaller microcells which can be reassigned to the available servers in order to distribute the load.

Dividing the world into more and smaller microcells does have its own drawbacks. As more cells exist, more communication between the microcells will occur as well. The total length of all borders will increase and so will the amount of cross-border interactions, which are typically more complex. However, the use of microcells does allow to use the available hardware more efficiently and makes it possible to cope with highly dynamic player migrations.

In [6] we presented an evaluation of the microcell concept and a number of algorithms to determine the optimal deployment of microcells. The results showed that improvements of up to 70% for the server load distribution can be obtained. In this paper we evaluate J2EE and JAIN SLEE as middleware solutions for implementing an MMOG with dynamic microcells.

3 Architecural overview

The global deployment architecture of an MMOG consists of much more than just the architecture of the actual software. There is a complex server infrastructure, often consisting of multiple tiers, to allow the massive network load to be handled and redirected as efficiently as possible. The microcell architecture (software architecture) is responsible for the actual game processing and data management.

3.1 MMOG Platform Architecture

The typical deployment architecture of a current MMOG consists of three or four tiers. First of all the players control the clients which display the players view on the virtual world (graphics) and the user interface. Next there is an infrastructure of firewalls, proxies and gateways to protect the game from attacks coming from the Internet. Another functionality often taken care of by this tier is the billing of the client and sometimes load balancing by redirecting the players connection to less loaded servers when logging in. Finally there are the actual game servers which process all incoming player actions, manage the game world and send responses back to the players. Usually these servers are backed up by a large database which holds the game content, the player inventories etc. This part of the architecture corresponds with the bottom half of figure 3. The game plane of the figure represents the microcells that hold all their data and the data of the players populating it.

3.2 Microcell Architecture

The actual microcell architecture is quite straightforward. All microcells deployed on one server will be under the control of a Microcell Controller. On each server holding microcells exactly one Microcell Controller will be deployed. This component takes care of the mobility of the microcells as it is responsible for the actual reassignment. Reassigning the microcells will be managed by the Microcell Manager, which will gather data on the server load and the load generated by the different components. If the Microcell Manager is able to calculate a better deployment it can initiate a relocation of all the related cells. The Microcell Controllers will then reassign the microcells and take care of all rerouting and forwarding of messages associated with this process.

The Microcell Manager is defined as a single logical component but could easily be implemented in a distributed fashion. For example a very basic distributed implementation could simply dispose of local microcells and reassign those to the least loaded neighbors.

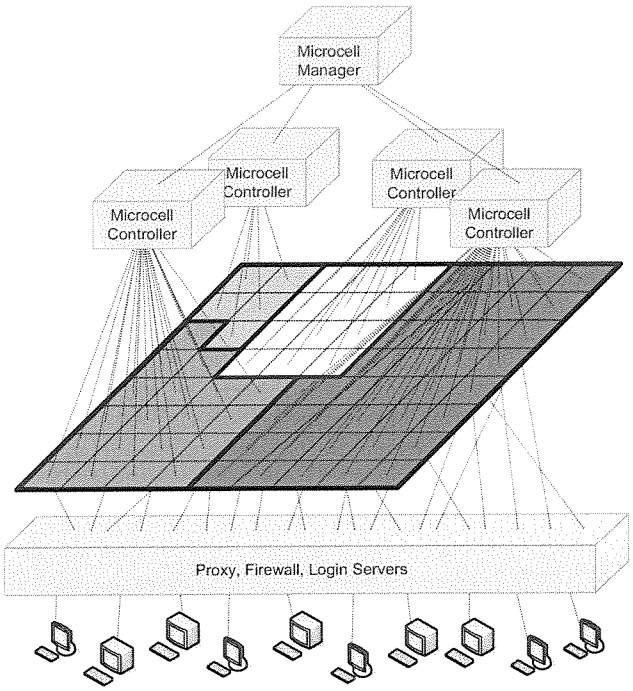


Fig. 3. Overview of the Microcell architecture. Every Microcell Controller is responsible for a set of Microcells. The Microcell Manager takes care of the actual placement of the Microcells. Clients connect to the game through a set of proxies, gateways and a firewall which protect the game servers from the network.

4 Requirements

The requirements that need to be met when developing and deploying an MMOG, as proposed in this paper, can be divided in two main categories. First there are the general considerations regarding latency and user experience, if the processing takes too long, the perceived latency will be noticeable and will eventually render the game unplayable. The proposed architecture should have as little influence on the actual game design as possible. It has to be completely transparent whether a microcell resides on a local server or on another server in the network.

4.1 Gameplay: Latency

The amount of latency that is acceptable varies from player to player and depends on the type of online game. When dealing with First Person Shooters (FPS) like Quake 3 the latency should preferably be lower than 100ms [7]. Taking into account the global network delay experienced on the Internet, this leaves a very limited time for processing. Other types of games like Online Role Playing Games (RPG) and Online Real Time Strategy (RTS) games usually have less strict requirements up to 500ms [8]. In FPSs the latency requirements are a lot more stringent as the direct interaction between users is much more important, especially for precise aiming. RTS games often allow the players to play independently for a large amount of time (e.g. gathering resources, constructing buildings, raising an army) before they actually start interacting with each other and the actual player interactions usually do not require the same amount of precision as FPSs, which all results in a higher tolerable latency.
As the virtual worlds in FPSs grow bigger with each generation of games and the interactions in RPGs and RTSs get more complex and require higher precision, we want to keep the Dynamic Microcell Architecture as general as possible, able to be used as a base for both (more and more overlapping) categories. Therefore the architecture may not be the limiting factor when it comes to latency. A good example of the merging categories is PlanetSide [9] which can be considered one of the first Massively Multiplayer Online Shooters.

4.2 Architecture

Looking from the architectural and design point of view, other requirements surface. The concept of moving microcells has to be completely transparent for both the users and the developers. It must not matter if a cell is located on the same server or not, interacting with a software component through the network or on the local machine has to be transparent.
Player messages will be sent to the game server, processed and then forwarded to other players. These messages are small isolated actions, that might cause other actions to happen in the game world. For example, when a player slashes his sword at a tree, the tree could fall down crushing another player in the process. This results in a large amount of messages which are best handled asynchronously. Hence the architecture needs to provide extensive support for message- or event-oriented communication, both internally and over the network.
The Dynamic Microcell concept basically divides the game world into different components, therefore it would simplify the actual implementation to use a component based infrastructure. This facilitates the mapping of the concepts to software components. As all microcells contain the data of their part of the world and the players populating it, this data has to be easily accessible and manageable.

5 Technologies

Building the Dynamic Micrcocell Architecture from scratch would be tedious and error prone. Therefore the use of an existing middleware platform has been investigated and evaluated.
J2EE and JAIN SLEE are both component based technologies and offer a container for the applications to be deployed and run in. Basically a container is an environment which takes care of the life cycle management (e.g. starting and stopping the application or application components) of the application

and allows that a number of non-functionals are delegated to the container in stead of the actual application. By non-functionals we mean features that are not strictly necessary for a correct functional application, but which are nevertheless very important. These features include logging, authentication, management of external resources, etc.

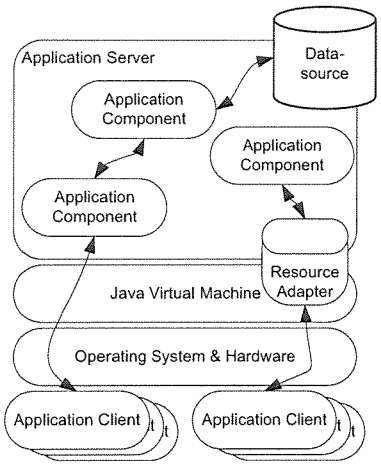


Fig. 4. Container based architecture which J2EE and JAIN SLEE have in common.

The global overview of a container managed framework is shown in figure 4. The Container can have multiple applications and/or application components deployed in the container. These components can interact with each other and other data sources, such as a database. Clients may interact directly with the application or may interact with the application container through the use of "Resource Adapters" (RA) which manage communication with the outside world.

5.1 J2EE

A typical J2EE Application Server consists of both an application container and a web container. The web container hosts all web related application components such as HTTP Servlets, Java Server Pages etc. The application container hosts J2EE applications composed of Enterprise Java Beans. There are three types of EJBs: Session Beans which usually contain business logic, Entity Beans which represent data and Message-Driven Beans which allow asynchronous processing and communication.

J2EE supports communication with clients through its web container using HTTP, or application clients can interact directly with application components using a J2EE application client which performs Remote Method Invocations. Apart from this there is also support for deploying J2EE Connector Architecture (JCA) [10] Resource Adapters into the application server. Through the use of these RAs, which can be deployed on any J2EE application server, it is possible to extend the application server and support extra methods of communication. A well known example using the JCA is the Java Message System (JMS) [11]. The J2EE Connector Architecture was originally designed for implementing Resource Adapters to interface with existing (legacy) systems, but it can be used in a more general context to extend the J2EE application server with extra protocol stacks.

Functionally, J2EE fulfills all the addressed requirements. It makes an abstraction of the network and there is hardly any difference between accessing a local component or a remote component. It is component based which allows us to easily map the Dynamic Microcell Architecture onto logical J2EE components. And very important, it features a versatile data management infrastructure needed for the microcells. Furthermore it is possible to extend the Application Server with Resource Adapters which allow us to use any custom communication protocol if desired. One possible disadvantage of J2EE is the limited possibility for asynchronous event handling.

5.2 JAIN SLEE

The JAIN SLEE specification, defined in JSR 22 [12], provides us with an application server tailored for telecom and telecom like applications (i.e. applications with strict low latency requirements). The Service Building Block (SBB) is the base component for building applications and can be considered the equivalent of EJBs in J2EE. One of the key features of the JAIN SLEE application container is the use of asynchronous communication. Internally almost all communication in the SLEE happens through the use of events. The SLEE uses the publish-subscribe model for event distribution. Communication through "simple" method calls is still possible if an SBB has defined a local interface. In that case, other SBBs could perform regular synchronous method calls. The idea behind the event based communication between SBBs is that every SBB performs its own task and then hands the result off to the next SBB in line. One could compare this to an assembly line in a factory

The internal routing of the events is completely taken care of by the application server. A key component in the routing of events is the Activity Context which manages the links between logically connected SBB entities. When an initial event enters the SLEE, an Activity Context is created. The SBBs processing this event will be attached to this Activity Context and all following events which logically belong together, will be fired on the same Activity Context and processed by the same SBB entity. This is important as the SBB entities can share data on this Activity Context.

Communication with the outside world happens only through pluggable Resource Adapters (RAs). Because of this JAIN SLEE is protocol agnostic, as any protocol can be supported by adding an appropriate RA with support for the desired protocol. The RA will process any incoming messages and convert them to events which can be handled by the application server. This does not necessarily mean that every application will be protocol agnostic, but certain components (e.g. a billing service) can be.

Functionally, JAIN SLEE meets all desired requirements, just like J2EE. The core of JAIN SLEE is event based, which

suits the needs of an MMOG with a large amount of external events that need to be processed. Furthermore, JAIN SLEE is specifically designed to handle applications with very low latency requirements. One drawback is the limited possibility for managing large amounts of data typical for MMOGs.

6 Benchmark Setup

To benchmark the application servers we consider the common use case for the MMOG of one player taking an action which has consequences for other players nearby. However, to avoid having to implement the MMOG architecture first and possibly conclude that the responsiveness of the application server is too low, we mapped the use case onto an existing scenario for setting up a Voice over IP (VoIP) call with similar time critical requirements. The characteristics of players sending and accepting messages in an MMOG are practically identical to those of people setting up a VoIP call using the Session Initiation Protocol (SIP). As benchmarks for SIP testing are publicly available and an implementation for the SIP scenario was already available for JAIN SLEE it was simply a matter of converting the existing application to J2EE.

6.1 Use Case

A graphical representation of the selected use case is shown in figure 5. One player (Alice) takes an action, for example firing a gun. The game client will transfer this event to the game server where it is processed by the listening resource adapter (1). Once accepted by the resource adapter, the player action will be transformed into an event which is then forwarded to the microcell Alice is residing in (2). The microcell will process the action and it will notice that the bullet coming from Alice's gun will fly right into one neighboring microcell and the actual shot will be audible in another microcell (3).

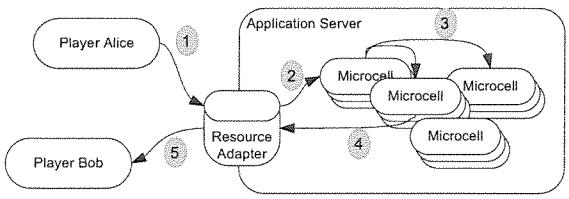


Fig. 5. Selected use case for the benchmark.

This new event will be received by the new microcell which in turn will process the event by calculating the course of the bullet. As it appears, player Bob happened to be in the wrong place at the wrong time and is hit in the chest. This notification is sent to the resource adapter by the microcell (4). The resource adapter turns this event into a transmittable message which is sent to player Bob who's client receives the unfortunate news of the hit and resulting wound (5).
The use case described is very similar with the use case of setting up a Voice over IP call using a SIP Proxy. Both use cases feature the sending and receiving of messages and processing by a central component (either the microcell or the proxy). As a SIP Proxy implementation was available for JAIN SLEE, including a JAIN SLEE SIP Resource Adapter, we implemented an equivalent implementation, based on the same proxy source code, for J2EE. J2EE does not have support for SIP out of the box, therefore we wrapped the publicly available JAIN SIP reference implementation [13] in a Resource Adapter implementation. The RA accepts incoming SIP messages and turns these into events which can be processed by Message-Driven Beans which process the message and forward it to the appropriate destination.

6.2 SIP

The Session Initiation Protocol (SIP) is defined in RFC 3261 [14] and describes an application-layer control (signaling) protocol for creating, modifying and terminating sessions with one or more participants. These sessions include Internet telephone calls, multimedia distribution, and multimedia conferences. RFC 3261 also defines the use of SIP proxies and how they should interact with other SIP applications and proxies.
The scenario used for the testing and benchmarks is the Proxy 200 test shown in Figure 6 defined in the SIPstone benchmark [15]. We are interested in the time it takes to set up a call, this is the time it takes from the initial INVITE of Alice till she receives an OK from Bob, indicating the call is set up. After this Alice will send an acknowledgment to Bob saying she received the OK and the media session (e.g. voice or video conference) can start. When benchmarking the call was immediately terminated by the caller and no media session was initiated.

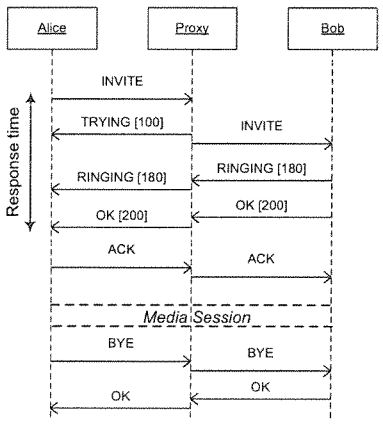


Fig. 6. Proxy 200 test

When of setting up a VoIP call, the time required for crossing one node (e.g. a VoIP SIP proxy) should not exceed 50ms for the 95th percentile of the calls and should be below 25ms for the 50th percentile.

7 Test Setup

Before we discuss the obtained results, a short overview of the test setup used is presented.

7.1 Software Setup

For benchmarking purposes SIPp [16] is used. SIPp is a free Open Source test tool/traffic generator for the SIP protocol. It allows generating SIP traffic and to establish and release multiple calls. It can also read custom XML scenario files, describing from very simple to complex call flows. It includes a few basic SIPstone defined test setups. The tests presented in this paper were performed using SIPp and the previously specified scenario.
The evaluated application servers where JBoss [17] for J2EE and Open Cloud Rhino [18] for JAIN SLEE.

7.2 Hardware Setup

All tests were performed using a dual Opteron 242 HP DL 145 with 2GB of memory for the proxy. The clients were run on AMD athlonXP 1600+ machines with everything interconnected in a 100Mbit switched ethernet network. All platforms were running Debian GNU/Linux with a 2.6 kernel and the Sun JDK 1.4.2.

7.3 Platform Tuning

For all the presented test results extensive tuning of the JVM and garbage collector was performed. Without tuning of the JVM the garbage collector initiated pauses in the execution of the application which resulted in calls timing out, even at low call rates. With appropriate tuning, these pauses can be minimized and thus the obtained results can be significantly improved. The basic set of tuning options used for all platforms is shown in table I. Detailed results of JVM tuning were previously reported on in [19], [20].

-Xmx512m
-XX:+UseParNewGC
-XX:+UseConcMarkSweepGC
-XX:+CMSIncrementalMode
-XX:+CMSIncrementalPacing
-XX:CMSIncrementalDutyCycleMin=0
-XX:CMSIncrementalDutyCycle=10
-XX:+UseTLAB
-XX:MaxTenuringThreshold=0
-XX:SurvivorRatio=128

TABLE I
VIRTUAL MACHINE TUNING OPTIONS FOR LOW LATENCY BEHAVIOR.

The options specified in table I are specific to the Sun JVM, but other Virtual machines offer similar tuning options which can be used to achieve the same effect. All tests were performed using the same JVM.

8 Evaluation Results

This section gives an overview of the obtained test results. Both platforms were submitted to a number of subsequent test runs that allowed us to evaluate and interpret the obtained data.
Figure 7 shows the performance results of the tested J2EE application server. The graph shows the average response time, the 50th percentile and the 95th percentile plotted against the average cpu load at the given call rate. As all tests were performed on a dual cpu machine the cpu load is the average of the load of the two cpus during the test.
During each test run every call rate, starting at 10 calls per second (caps) was run for 5 minutes. Then there was a pause of 1 minute after which the subsequent call rate was tested. The call rates were increased for as long as the system under test could sustain the tested call rate. Each test run was repeated three times to check consistency among the different test runs. Before every test run, a dummy run was performed as well to allow the JVM to "warm up".

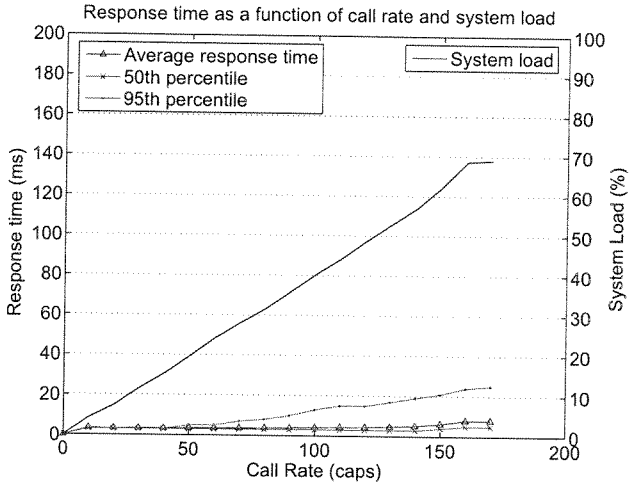


Fig. 7. Performance results of the J2EE Proxy 200 test.

Before we explain the results, it is important to note that the load on the two cpus was not equally distributed during the J2EE test runs. We assume this is due to limitations of the SIP stack used which is a reference implementation, not optimized for production use. This is also the reason why the cpu load does not increase above 70% as the SIP Stack is overloading one cpu. This did not occur when evaluating JAIN SLEE. J2EE was able to sustain the call rate up to 150 caps on the specified hardware. At higher call rates a number of calls time out due to lack of a high performance SIP stack.
The results for JAIN SLEE are shown in figure 8. JAIN SLEE was able to sustain a call rate of 200 caps without any timeouts occurring. At this point the cpu load reaches almost 100% and it is no longer possible to increase the call rate.
When comparing the results of J2EE and JAIN SLEE it is clearly shown they both meet the strict low latency requirements for setting up a SIP call. At a call rate of 100 caps both technologies realize response times of less than 10ms, this means that approximately 1300 messages per second can be processed at a cpu load of less than 50%. If we look at the original use case of one player generating an event which results in another event for a second player the achieved response times of less than 10ms still leaves extra room to allow for more complex processing if necessary and

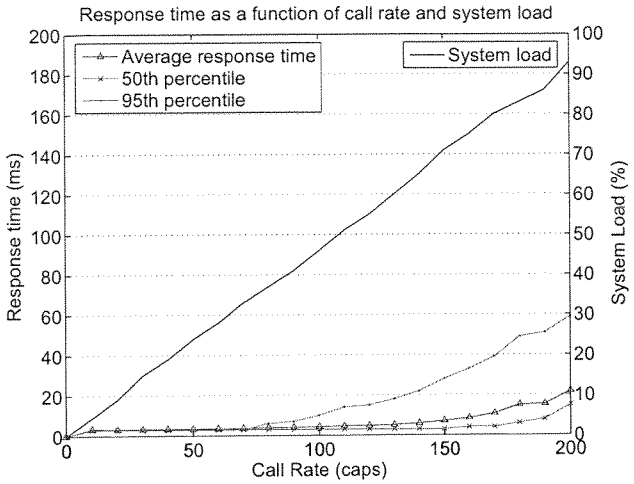


Fig. 8. Performance results of the JAIN SLEE Proxy 200 test.

additional network delay that will occur on the Internet, to still meet the requirement of a 100ms latency. Thus, based on the performance results both technologies could be used to serve as MMOG middleware.

9 Conclusions and Future Work

In this paper, a novel architecture for an MMOG platform was presented, designed to cope with varying load distributions by dynamically redeploying the game world over the available servers. In order to implement the platform we first evaluated two existing Java technologies, J2EE and JAIN SLEE, both functionally and performance-wise. Functionally both platforms meet all the necessary requirements, they offer a component based architecture and make an abstraction of the network. Regarding data management, we observe that J2EE has the most extensive support out of the box, whereas the support in JAIN SLEE is rather limited. The event based architecture found in JAIN SLEE on the other hand, is tailored specifically for this type of applications with a large amount of events and low latency requirements. Considering the performance results, J2EE and JAIN SLEE offer similar results and are both capable of low latency behavior. This makes both platforms a suitable choice for implementing the Dynamic Microcell Architecture.

One area that needs more research is the design of algorithms to determine the "optimal" microcell deployment. A number of algorithms were already proposed and evaluated in [6]. However, these algorithms do not take the current deployment into account. Therefore, in order to completely take advantage of the dynamic nature of the microcell deployment, a new set of algorithms is needed that try to find a balance between the performance of the new deployment and the effort needed to redeploy the microcells (e.g. the number of microcells to be moved). In order to validate the usability of the proposed architecture, an implementation of the platform is needed. This can then be used to test the performance of the architecture under varying loads, to improve the deployment algorithms, etc.

Acknowledgment

We would like to thank OpenCloud for providing us with a license for their JAIN SLEE implementation Rhino.
The research in this paper is partially funded by the IBBT T-Case project.

References

[1] Blizzard Entertainment Press Release, "World of warcraft achieves new milestone with two million paying subscribers worldwide," [online], 2005, http://www.blizzard.com/press/050614-2million.shtml.

[2] P. Rosedale and C. Ondrejka, "Enabling player-created online worlds with grid computing and streaming," *Gamasutra*, September 2003.

[3] Microsoft, "Xbox 360 fact sheet," [online], 2005, http://www.xbox.com/en-US/xbox360/factsheet.htm.

[4] Nintendo press release, "Nintendo's compact console will turn the world of gaming on its side," [online], 2005, http://www.nintendo.com/newsarticle? articleid=02ea1a40-ac09-4cdf-9548-91e5a4e78746&page=other.

[5] Sony press release, "Sony to launch its next generation computer entertainment system," [online], 2004, http://www.us.playstation.com/Pressreleases .aspx?id=279.

[6] B. De Vleeschauwer, B. Van Den Bossche, T. Verdickt, F. De Turck, B. Dhoedt, and P. Demeester, "Dynamic microcell assignment for massively multiplayer online gaming," in *Proceedings of Netgames 2005: 4th Workshop on Network and Systems Support for Games*, New York, USA, 2005.

[7] G. Armitage, "An experimental estimation of latency sensitivity in multiplayer Quake 3," in *Proc. of the 11th IEEE International Conference on Networks (ICON2003)*.

[8] N. Sheldon, E. Girard, S. Borg, M. Claypool, and E. Agu, "The effect of latency on user performance in Warcraft III," in *Proc. of ACM Network and System Support for Games (NetGames)*, May 2003, pp. 3–14.

[9] Sony, "Planetside website," [online], http://planetside.station.sony.com/.

[10] Sun Microsystems, "J2EE Connector Architecture 1.5," [online], http://www.jcp.org/en/jsr/detail?id=112.

[11] M. Hapner, R. Burridge, R. Sharma, J. Fialli, and K. Haase, *Java Message Service API Tutorial and Reference: Messaging for the J2EE Platform*. Addison-Wesley Professional, 2002.

[12] Sun Microsystems, "JAIN SLEE API Specification," http://jcp.org/en/jsr/detail?id=22.

[13] M. Ranganathan and P. O'Doherty, "jain-sip: JAVA API for SIP Signaling," [online], https://jain-sip.dev.java.net/.

[14] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J.Peterson, R. Sparks, M. Handley, and E. Schooler, "Session Initiation Protocol," [online], 2002, http://www.ietf.org/rfc/rfc3261.txt?number=3261.

[15] H. Schulzrinne, S. Narayanan, J. Lennox, and M. Doyle, "SIPstone - Benchmarking SIP Server Performance," [online], April 2002, http://www.sipstone.org/.

[16] Hewlett-Packard, "SIPp : SIP benchmarking utility," [online], http://sipp.sourceforge.net/.

[17] JBoss Inc., "JBoss Application Server," [online], http://jboss.com/products/jbossas.

[18] Open Cloud, "Open Cloud Fault Tolerant and Carrier Grade, Middleware and Application Server Products," [online], http://www.opencloud.com/.

[19] B. Van Den Bossche, F. De Turck, B. Dhoedt, T. Pollet, B. Van Vlerken, J. Moreels, N. Janssens, P. Demeester, and D. Colle, "Evaluation of Current Java Technologies for Telecom Backend Platform Design," in *Proceedings of the 2005 International Symposium on Performance Evaluation of Computer and Telecommunication Systems*, July 2005, pp. 699–709.

[20] B. Van Den Bossche, F. De Turck, B. Dhoedt, and P. Demeester, "Enabling Java-based VoIP backend platforms through JVM performance tuning," 2006, to be published in the proceedings of The 1st IEEE workshop on VoIP Management and Security: VoIP MaSe co-located with IEEE NOMS 2006.

ICOMP'06

The 2006 International Conference on Internet Computing

Session: APPLICATIONS

Session: TRUST AND SECURITY

Zhan Liu, Mi Lu

Session: WEB–SERVICES, INTERFACES AND LANGUAGES

Language Model Grammar Conversion

Wesley Holland, Julie Baca, Dhruva Duncan, Joseph Picone

Developing Interoperable Software For Differing XML Schemata

Thuy–Linh Nguyen

Techniques for Handling JSF Exceptions, Messages and Contexts

Dwight Deugo

Index and Search XML Documents by Combining Content and Structure

Faïza Abbaci, Jean–Baptiste Valsamis, Pascal Francq

A Distributed API for Searching Multimedia Databases

Dale Parson, Lisa Frye

Automatic Web Service Composition Based on Service Interface Description

Jing–zhou Zhang, Shou–jian Yu, Xiao–kun Ge, Guo–wen Wu

A Framework for Collaborative Applications using Web Services

Ricardo Coppo, Claudio Delrieux

Session: USABILITY ASPECTS AND VISUALIZATION

A Methodology for Structured Use–Centered Quantitative Full–Life–Cycle Usability Requirements Specification and Usability Evaluation of Web Sites

Guoqiang Hu, Kai H. Chang

The Early Stages of TA Spoken Shared Internet Browser for the Blind

Emad Eldin Mohamed, Hesham Kamel

A Study on the Screen Designs of E–Learning Material

Hiroshi Ichikawa, Kaoru Honda, Hiroo Hirose, Yoshito Yamamoto

Proposal of Web usability by comparison experiment of paper media and internet browser that considers individual characteristics

Hiroo Hirose, Hiroshi Ichikawa, Yoshito Yamamoto

Can We Improve Web Accessibility for Users with Below Average Cognitive Capacity?

Jenny Jerrams–Smith, David Heathcote

Session: SENSORS, NETWORKS AND COMMUNICATION

Hybrid WDM and Optical Spectral Amplitude Coding with Array Waveguide Gratings over Fiber–to–the–Home Network

Che–Chih Hsu, Jen–Fa Huang, Yao–Tang Chang

A Web–based Server Management System with IPMI and WMI Techniques

Yi–Hsuan Yeh, Yu–Chin Szu, Yuan–Cheng Lai

Distributed Sensor Networks based on Hybrid Communication Model

Mohammed Ketel

Session: QOS AND TRAFFIC MANAGEMENT

Session: ALGORITHMS, EVALUATIONS AND MEASURMENTS

Session: 4TH WORKSHOP ON XML TECHNOLOGY AND APPLICATIONS – XML TECH'06

Efficient Filtering System to accelerate XML Access Control Enforcement

Changwoo Byun Byun, Seog Park

On–Line News Management System Based on ASP and XML

Chen Hua, Tang Chunyan, Yang Yuexin, Cai Yan, Fang Zhoui, Zhao Rujia, Joan Lu

Virtual Enterprise Oriented Enterprise Strategies and Cooperative Product Design System Development Based on J2EE MVC Mode

Cao Yan, Chen Hua, Yang Lina, Liu Ning, Zhao Rujia, Joan Lu

The Application of XML in the Denotation of Plant Diseases and Insect Pests Information

Qingming Fan, Hua Chen, Yan Cao, Baolong Liu, Yuexin Yang, Bo Sun, Rujia Zhao, Joan Lu

Semantic Enhanced Self–Configuring Grid Framework

Hao Li, Gehao Lu, Joan Lu, Shaowen Yao

Examining Analysis and Evaluation System Based on XML

Chen Hua, Lv Qingli, Cao Yan, Liu Baolong, Yuan Li, Zhao Rujia, Joan Lu

XQuery as a Spatial Query Language

Xia (Lisa) Li

The Application of XML Parsing Technology in E–Government

Gang Xue, Lili Zhang, Rujin Yang, Hao Li, Shaowen Yao

XML Security in Certificate Management

Joan Lu, Nathan Cripps, Hua Chen

Agent Mediated SOA with XML Framework for Grid Computing

Gehao Lu, Hao Li, Joan Lu, Shaowen Yao

XML–Based Security – A New Challenge to I.T. Protection

J. Lu, U. Rahman, Jim Yip

Session: MOBILE MULTIMEDIA + WIRELESS CONTENT DELIVERY ISSUES

Modelling Service Delivery System for Wireless Handsets using Instant Alert and Frequent Scanning Technologies: An Approach for Bus Service Application

Johnnes Arreymbi, Abdullah Al–Zakwani

Mobile Devices Evolution and Revolution: A Cause for Security Concern

Abdullah Al–Zakwani

Interactive Design and Delivery Challenges for Wireless Handheld Multimedia Systems

Johnnes Arreymbi, Esther Gachanga

Session: INTERNET COMPUTING TOOLS AND ANALYSIS

Tor – A Tool to Disguise Internet Communications

Victor Clincy, Padmaja Mudiraj

Session: PROCEEDINGS OF CONFERENCE ON COMPUTER GAMES DEVELOPMENT

The 2006 World Congress in Computer Science, Computer Engineering, and Applied Computing

Monte Carlo Resort, Las Vegas, Nevada, USA
June 26–29, 2006

Conferences:

The 2006 International Conference on Bioinformatics & Computational Biology

The 2006 International Conference on Computer Design & International Conference on Computing in Nanotechnology

The 2006 International Conference on Computer Graphics & Virtual Reality

The 2006 International Conference on Communications in Computing

The 2006 International Conference on Scientific Computing

The 2006 International Conference on Data Mining

The 2006 International Conference on e–Learning, e–Business, Enterprise Information Systems, e–Government, & Outsourcing

The 2006 International Conference on Engineering of Reconfigurable Systems & Algorithms

The 2006 International Conference on Embedded Systems & Applications

The 2006 International Conference on Foundations of Computer Science

The 2006 International Conference on Frontiers in Education: Computer Science & Computer Engineering

The 2006 International Conference on Grid Computing & Applications

The 2006 International Conference on Artificial Intelligence

The 2006 International Conference on Internet Computing & International Conference on Computer Games Development

The 2006 International Conference on Wireless Networks

The 2006 International Conference on Information & Knowledge Engineering

The 2006 International Conference on Image Processing, Computer Vision, & Pattern Recognition

The 2006 International Conference on Machine Learning: Models, Technologies & Applications

The 2006 International Conference on Modeling, Simulation & Visualization Methods

The 2006 International Conference on Parallel & Distributed Processing Techniques & Applications & International Conference on Real-Time Computing Systems & Applications

The 2006 International Conference on Pervasive Systems & Computing

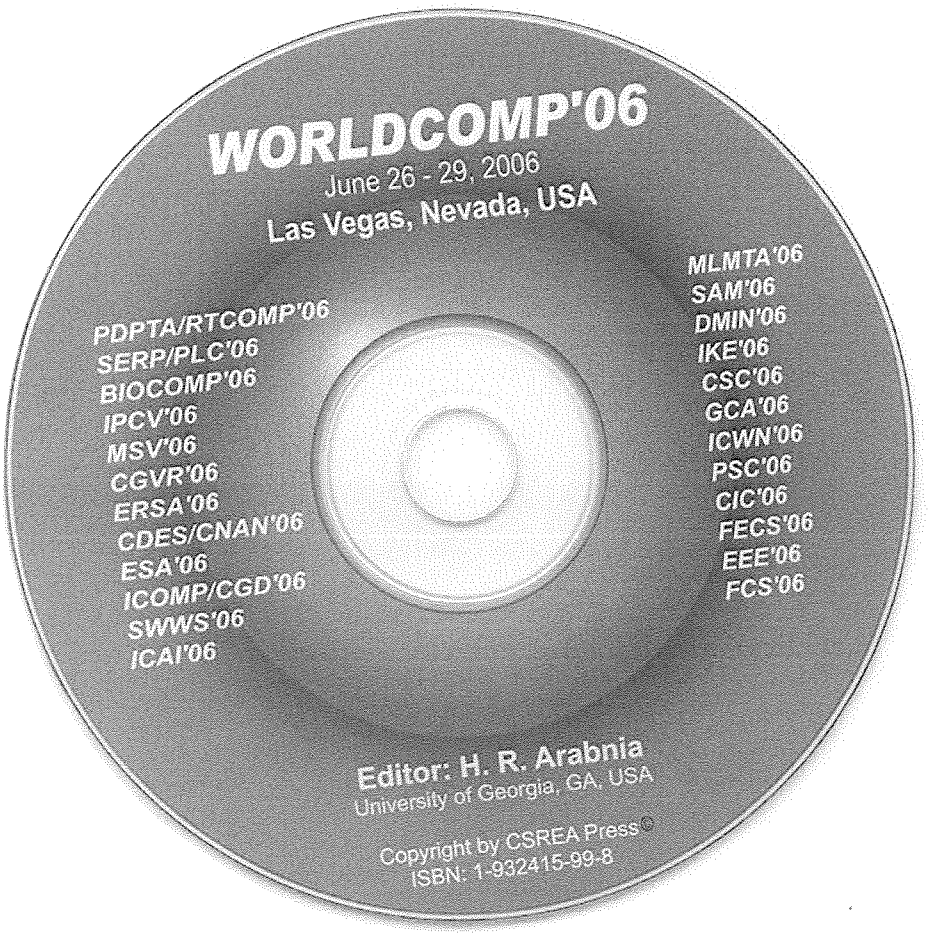
The 2006 International Conference on Security & Management

The 2006 International Conference on Software Engineering Research & Practice & International Conference on Programming Languages and Compilers

The 2006 International Conference on Semantic Web & Web Services

Editor H.R. Arabnia
University of Georgia, GA, USA

ISBN: 1-932415-99-8


WORLDCOMP'06
June 26 - 29, 2006
Las Vegas, Nevada, USA
PDPTA/RTCOMP'06
SERP/PLC'06
BIOCOMP'06
IPCV'06
MSV'06
CGVR'06
ERSA'06
CDES/CNAN'06
ESA'06
ICOMP/CGD'06
SWWS'06
ICAI'06
MLMTA'06
SAM'06
DMIN'06
IKE'06
CSC'06
GCA'06
ICWN'06
PSC'06
CIC'06
FECS'06
EEE'06
FCS'06
Editor: H. R. Arabnia
University of Georgia, GA, USA
Copyright by CSREA Press©
ISBN: 1-932415-99-8